Approximating Certainty in Querying Data and Metadata

Approximating Certainty in Querying Data and Metadata

Cristina Civili and **Leonid Libkin**
University of Edinburgh

Abstract

Metadata, such as mappings or constraints, is used in a variety of scenarios to facilitate query answering; these include data integration and exchange, consistent query answering, and ontology-based data access. A common feature of these scenarios is that data and metadata together produce multiple databases, and answers to queries must be certain, i.e., true in all such databases. This usually incurs prohibitively high complexity outside very restricted classes of queries such as conjunctive queries and their unions.

To overcome this, we propose to approximate such query answering by reducing it to another scenario where multiple databases need to be taken into account, namely incomplete information in databases. For them, well-behaved approximation schemes exist for much larger classes of queries. We give a generic representation of query answering via incomplete data, and show how it works in the scenarios listed above. We use the connection to show how to effectively approximate several intractable query answering problems, and discuss differences between applying this framework under open and closed world semantics.

1 Introduction

There are many applications where reasoning about data and metadata is essential for providing meaningful answers to queries. These include data integration (Lenzerini 2002), data exchange (Arenas et al. 2014), ontology-based data access (OBDA) (Bienvenu and Ortiz 2015; Poggi et al. 2008), and consistent query answering (CQA) (Bertossi 2011). They all follow the same pattern: using existing data together with some additional knowledge provided as metadata, they define a set of possible – but unknown to us – datasets, and then attempt to find query answers that are guaranteed to be true in all of them. For example, in data integration and data exchange, existing data is information we have in data sources, and metadata is provided by the schema mapping. Together, they define multiple databases of global or target schema, and query answering has to take all of them into account. In consistent query answering, one takes an inconsistent database and constraints as metadata, and looks at repairs that restore consistency. In OBDA, a database, a mapping, and an ontology together define the set of all database extensions consistent with the ontology; query answering must take all of them into account.

In all these applications, query answering must extract *certain information* from many databases. This can only be done efficiently for a rather restricted class of queries. Indeed, to check a query in many databases, we essentially need to solve a version of the validity problem, and such problems are hard. As a result, a frequent assumption underlying most of the above scenarios is that only unions of conjunctive queries (UCQs) are considered. There is very little known about answering queries beyond UCQs; it is viewed as being very costly.

There is another area that faces a very similar problem, namely handling of incomplete information in databases. An incomplete database represents many (perhaps infinitely many) complete ones, and query answers must be certain, i.e., consistent with all of them. Similarly, only queries such as UCQs can be answered efficiently with certainty (Imielinski and Lipski 1984). However, in the case of incomplete information, we know that for many more queries certain answers can be *approximated* (e.g., all those definable in first-order logic, under the closed world semantics of incompleteness (Guagliardo and Libkin 2016)).

Our idea then is to leverage this knowledge of how to efficiently approximate answers to queries over incomplete databases, and apply it to the scenarios where additional knowledge assists query answering, towards the goal of providing answers – even partial ones – to queries that go beyond UCQs. Specifically, for each of the application scenarios where we have a database D, some metadata Σ, and a query Q:

1. we first build a single incomplete database D' that represents all (and perhaps more) datasets that arise from D and Σ combined;
2. then we find an approximation to answers to query Q on that database D', and show that it will be guaranteed to provide an approximation in our application scenario.

Here and elsewhere, by approximating a query we mean finding a subset of its answers, that is, finding some of the query answers that are guaranteed to be correct.

The reason we cannot simply run the original query Q over the incomplete database D' that the first step produces is that this may return false answers for queries outside the

Figure 1: Source data and canonical solution for Example 1

Source:

UnpaidOrder

oid	cid	total
02	771	46
03	771	31

ScheduledPaymentOrder

oid	cid	total	pdate
01	894	60	9/24/18

PaidOrder

oid	cid	total
04	894	22

Client

cid	name	status
771	John	'gold'
894	Mary	'silver'

Canonical Solution:

Order

oid	cid	total
01	894	60
02	771	46
03	771	31
04	894	22

Payment

pid	poid	pcid	pdate
$\perp_1$	04	894	$\perp_2$
$\perp_3$	$\perp_4$	771	$\perp_5$
$\perp_6$	01	894	9/24/18

Client

cid	name	status
771	John	'gold'
894	Mary	'silver'

class of UCQs (we shall see an example shortly). As for the second step, of course there are many trivial approximation schemes (in the extreme case one can just return the empty set), but we shall see that better solutions exist. In fact we will show that genuinely useful approximations of this sort exist, for queries that go beyond UCQs and cannot be answered efficiently in application scenarios, and where furthermore running Q on the incomplete database D' produces wrong answers, but running the approximation to Q restores correctness and returns meaningful answers.

Let us illustrate this by an example based on a data exchange scenario.

Example 1. Consider a data exchange setting aimed at restructuring a database that stores data about orders and clients of a company. The source schema has relations UnpaidOrd(oid,cid,total) of unpaid orders with their ids, customer ids, and amount; ScheduledOrd(oid,cid,total,pdate) of orders scheduled for payment that also have the pdate attribute for payment date; PaidOrd(oid,cid,total) of paid orders, and Client(cid,name,status) of clients, with status indicating their standing with the company (gold, silver, etc.).

The company wants to restructure the data to keep track of payments and dates, and to enforce the rule that members with the 'gold' status have made at least one payment. This is reflected in the target schema with relations for orders: Order(oid,cid,total), for payments: Payment(pid,poid,pcid,pdate) (with payment, order, client ids, and dates), together with a copy of the Client relation.

Typically in data exchange and integration (Arenas et al. 2014; Lenzerini 2002), metadata comes in the form of source-to-target constraints known as tuple-generating dependencies; they are (abbreviating relation names in the obvious way):

$$\begin{aligned}
\mathsf{UO}(\mathsf{o},\mathsf{c},\mathsf{t}) &\rightarrow \mathsf{O}(\mathsf{o},\mathsf{c},\mathsf{t})\\
\mathsf{SO}(\mathsf{o},\mathsf{c},\mathsf{t},\mathsf{p}) &\rightarrow \exists z\ \mathsf{O}(\mathsf{o},\mathsf{c},\mathsf{t}) \wedge \mathsf{P}(z,\mathsf{o},\mathsf{c},\mathsf{p})\\
\mathsf{PO}(\mathsf{o},\mathsf{c},\mathsf{t}) &\rightarrow \exists z_1, z_2\ \mathsf{O}(\mathsf{o},\mathsf{c},\mathsf{t}) \wedge \mathsf{P}(z_1,\mathsf{o},\mathsf{c},z_2)\\
\mathsf{C}(\mathsf{c},\mathsf{n},\text{“gold”}) &\rightarrow \exists z_1, z_2, z_3\ \mathsf{P}(z_1,z_2,\mathsf{n},z_3)\\
\mathsf{C}(\mathsf{c},\mathsf{n},\mathsf{s}) &\rightarrow \mathsf{C}(\mathsf{c},\mathsf{n},\mathsf{s})
\end{aligned}$$

The first constraint copies all unpaid orders into Order, the second copies all scheduled orders and creates a payment record with an unknown id, the third copies paid orders and creates a payment record with unknown id and date, and fourth says that for each 'gold' customer there is a payment record, and the last one copies the Client table.

Figure 1 shows sample source data. Let Q be a query, over the target schema, asking for the ids of orders that will not be paid by today (expressed below in relational algebra):

$$\begin{aligned}
&\pi_{\mathsf{oid}}(\mathsf{O} - \pi_{\mathsf{oid,cid,total}}(\mathsf{O} \bowtie_{\mathsf{oid}=\mathsf{poid}} \mathsf{P}))\\
\cup\ &\pi_{\mathsf{oid}}(\sigma_{\mathsf{pdate}\geq\$\mathsf{today}}(\mathsf{O} \bowtie_{\mathsf{oid}=\mathsf{poid}} \mathsf{P}))
\end{aligned}$$

A simple reasoning based on analyzing information in the source shows that 01 is such an id that will certainly be in the answer, no matter what target instance we build, since it comes with a known order date (we assume it to be later than today). In data exchange, one answers queries over *one* particular target instance, most commonly the *canonical solution*. It is obtained by populating the target based on the rules, putting in null values for existential variables. It is shown in Figure 1, with nulls denoted by $\perp$.

Running Q on the canonical solution produces three answers: 01, 02, and 03. Of them, 02 and 03, are *false positive*, i.e., they are returned as answers to the query over the canonical solution, but they are not certain answers. In fact it has long been known that running the query over the canonical solution works only for unions of conjunctive queries (Arenas et al. 2014; Fagin et al. 2005) or small extensions thereof (Arenas, Barceló, and Reutter 2011).

Can we get rid of false positives? Notice that the canonical solution is a database with nulls, and for such databases, we know how to answer queries in a way that false positives are avoided, see e.g., (Guagliardo and Libkin 2016; Greco, Molinaro, and Trubitsyna 2017). But why would this be of any help? We are talking about different levels of certainty here: we need to produce answers that are true in all possible targets, but we only have techniques for approximating answers over one incomplete database. The reason it makes sense to approximate query answers on the canonical solution is a special property of it that, viewed as an incomplete database, it represents all possible target databases. This connection enables approximate query answering in data exchange via approximate query answering in incomplete databases.

This example illustrates points 1) and 2) made earlier: we can build an incomplete database instance that represents all databases we are interested in, and then approximate query answering over it, using techniques from query answering over incomplete data. The first of these steps is often easy to make in the applications such as data exchange, data in-

tegration, CQA, and OBDA. We refer to them as *application scenarios* from now on. For each of them, we can construct a specific incomplete instance that captures all the databases that a given application scenario generates (and perhaps more). This means that every approximation of certain query answering we can use from the incomplete information domain becomes usable in these application scenarios: it produces an approximation of query answers, and can be applied to much larger classes of queries for which previously no techniques existed.

In fact the process of building a database instance in step 1) is not our focus here as such instances, in most cases, are built by algorithms well known in the literature. Our main interest is in showing that they, together with approximation algorithms for querying incomplete information, can provide meaningful answers to queries that hitherto could not be handled in applications involving data and metadata (i.e., queries going beyond UCQs).

After formalizing this general technique, we show how it can be applied in data integration and exchange (using canonical solutions as incomplete databases), consistent query answering (using a special instance called canonical repair), and OBDA (using the result – even partial – of the chase). We give specific instances of approximations where nontrivial query answers without false positives can be produced for arbitrary first-order queries, in scenarios that were previously beyond reach.

Organization. Basic definition are in Section 2. Section 3 defines the notions of application scenarios. Section 4 shows how to reduce certain query answering in application scenarios to that over incomplete data. Section 5 and 6 use approximation schemes for first-order queries to find approximations of intractable querying problems in application scenarios. Concluding remarks are in Section 7.

2 Preliminaries

Relational databases A relational schema $\mathcal{R}$ is a set of relation names with associated arities. Database elements will come from a countably infinite set C of *constants*, which applications commonly supplement by inserting some unknown elements. These are called *nulls*; we assume that they come from a countably infinite set N, disjoint from C. A relational database instance, or just database, D over $\mathcal{R}$, is a finite set of atoms of the form $R(t_1, \ldots, t_k)$, where R is a k-ary relation in $\mathcal{R}$, and $t_1, \ldots, t_k \in \mathsf{C} \cup \mathsf{N}$. The set of all elements from $\mathsf{C} \cup \mathsf{N}$ that appear in atoms of D is called the *active domain* of D, denoted by $adom(D)$. A database D is *complete* if $adom(D) \subset \mathsf{C}$. The set of all databases over $\mathcal{R}$ is denoted by $Inst(\mathcal{R})$.

Given two relational instances D_1, D_2 over $\mathsf{C} \cup \mathsf{N}$, a *homomorphism* $h : D_1 \to D_2$ is a map from $adom(D_1)$ to $adom(D_2)$ such that $h(c) = c$ if $c \in \mathsf{C}$, and for each atom $R(t_1, \ldots, t_k)$ of D_1, the atom $R(h(t_1), \ldots, h(t_k))$ is in D_2. A homomorphism h whose range contains only constants is called a *valuation* (it provides values for nulls). Valuations are used to provide the semantics of a database D with nulls as $[\![D]\!] = \{v(D) \mid v \text{ is a valuation}\}$. In other words, an incomplete database D represents complete ones in which nulls are replaced by constants. This is known as the closed-world semantics (Reiter 1977); we shall deal with open-world semantics as well.

As our basic query language we take *first-order logic* (FO) over the relational schema. Its formulae are built up from relational atoms $R(\bar{t})$ and equality atoms $t = t'$ by using conjunction, disjunction, negation, and existential and universal quantification. We also look at fragments such as *conjunctive queries (CQs)* that only allow existential quantifiers and conjunction, and *unions of conjunctive queries (UCQs)* that also allow disjunction.

Some of the queries will be presented in relational algebra which of course is equivalent in expressiveness to FO, and is used to implement declarative queries in relational databases. The reason is that approximation schemes for finding query answers over incomplete databases are often presented at the level of relational algebra, and presenting queries in it allows us to skip the extra step of translating between languages.

Certain and correct answers In the scenarios we consider, a common feature of query answering is that answers must be consistent with a *family* $\mathcal{D}$ of databases, for instance, $[\![D]\!]$ for an incomplete database D. To answer a query Q, one therefore needs to extract certain information from $Q(\mathcal{D}) = \{Q(D) \mid D \in \mathcal{D}\}$.

If Q returns a set (e.g., of tuples, as for relational queries), one often defines certainty as $\Box_\cap(Q, D) = \bigcap_{D' \in [\![D]\!]} Q(D')$, see (Imielinski and Lipski 1984), or alternatively as $\Box_\perp(Q, D) = \{\bar{a} \mid v(\bar{a}) \in Q(v(D))$ for every valuation $v\}$, see (Lipski 1984). The latter has the advantage of also producing certain tuples that contain nulls, and $\Box_\cap(Q, D)$ is simply the set of tuples without nulls that occur in $\Box_\perp(Q, D)$.

Since our goal is to reason about *approximations* of query answers, and we do not want to be tied to one particular definition, we adopt a generic approach that defines certainty based on *relative informativeness* of query answers (Libkin 2016). The idea is that we have a relation $\preceq$ on objects that can appear as query answers, with $X \preceq Y$ meaning that Y is at least as informative as X; this relation is a preorder, i.e., reflexive and transitive.

Definition 1 (Correct and certain answers)**.** *Let $\mathcal{D}$ be a set of databases of the same schema $\mathcal{R}$, let Q be a query over $\mathcal{R}$, and let $\preceq$ be the relative informativeness ordering on query answers. Then an object X is a* correct answer *to Q over $\mathcal{D}$ if $X \preceq Y$ for each $Y \in Q(\mathcal{D})$, and it is a* certain answer *to Q on $\mathcal{D}$ if it is a maximal element, with respect to $\preceq$, among correct answers.*

If $\mathcal{D}$ and $\mathcal{D}'$ are sets of databases then $Q(\mathcal{D}) \sim Q(\mathcal{D}')$ means that correct answers to Q are the same on $\mathcal{D}$ and $\mathcal{D}'$.

Thus, if a certain answer exists, it is the greatest lower bound of all answers $Q(D')$ for $D' \in \mathcal{D}$, with respect to $\preceq$. For queries over relational data that return sets of tuples, it is standard to use the following definition: $X \preceq Y$ iff there exists a homomorphism $h : X \to Y$. Indeed, a homomorphism provides additional information by replacing nulls by constants, or equating nulls, or adding more tuples to the answers, thus giving a better approximation to correct answers.

If Q is a relational query over incomplete databases that returns a set of tuples, then both $\Box_\perp(Q, D)$ and $\Box_\cap(Q, D)$ are correct answers to Q over $[\![D]\!]$, with respect to $\preceq$.

3 Query Answering in Application Scenarios

We now provide a definition that formally captures scenarios of query answering assisted by additional knowledge provided as metadata. Essentially, we start with a data set and metadata (often given as a set of rules), and use those to produce (physically or virtually) multiple new data sets.

Definition 2. *An* application scenario *is modeled as a tuple $\mathcal{A} = \langle R_s, R_t, (\!|\cdot|\!)\rangle$ that consists of two relational schemas R_s and R_t, and a function $(\!|\cdot|\!)$ that takes an instance of R_s and produces a set of instances of R_t.*

Following the tradition of data integration and exchange scenarios, we call R_s and R_t *source* and *target* schemas (this is just a naming convention: in other cases this need not be their exact interpretation). Metadata may appear as a set of rules in data integration and exchange, or a set of constraints for CQA, or an ontology for OBDA. It is captured by the function $(\!|\cdot|\!)$, as will be explained shortly.

In the cases we consider, instances of R_t are arbitrary databases, and instances of R_s are complete. This is in line with the applications considered here, but is not a hard restriction and it does not affect the results. In fact in some cases, like data exchange, dealing with incomplete source instances has been studied (Arenas, Pérez, and Reutter 2013).

Given an application scenario $\mathcal{A}$ and a database instance $D \in \mathit{Inst}(R_s)$, we define the *support* $\mathcal{A}(D)$ of D under $\mathcal{A}$ as the set of all complete databases that represent databases in $(\!|D|\!)$, i.e., $\{v(D') \mid D' \in (\!|D|\!) \text{ and } v \text{ is a valuation}\}$. In other words, if we have incomplete databases in $(\!|D|\!)$, as often happens, the support expands $(\!|D|\!)$ by adding all possible interpretations of nulls in them.

We shall see, in Section 4, how instances of query answering appear as these application scenarios. Finding query answers then boils down to finding certain answers over supports $\mathcal{A}(D)$. These are often infeasible, and then one needs to approximate them by finding correct answers. We next explain the key idea behind finding such approximations.

Approximate Query Answering

The idea of approximating queries in application scenarios is to overapproximate their supports via incomplete databases, and then use our knowledge of how to approximate query answering over incomplete data, i.e., how to find correct answers to such queries. We consider two semantics of incompleteness. The CWA semantics (i.e., the semantics under the *closed-world assumption*), was introduced earlier; it is denoted as $[\![D]\!]^{\text{CWA}}$ or simply $[\![D]\!]$. The OWA semantics (under the *open-world assumption*), is defined as $[\![D]\!]^{\text{OWA}} = \{v(D) \cup D' \mid v \text{ is a valuation and } D' \text{ is complete}\}$. It extends the closed world semantics by adding arbitrary sets of complete tuples to databases.

Definition 3. *Given a semantics $[\![\cdot]\!]^*$ of incomplete data, an application scenario $\mathcal{A} = \langle R_s, R_t, (\!|\cdot|\!)\rangle$, and an instance $D \in \mathit{Inst}(R_s)$, we say that $D' \in \mathit{Inst}(R_t)$ is an $\mathcal{A}$-*cover *of D under $[\![\cdot]\!]^*$ if $\mathcal{A}(D) \subseteq [\![D']\!]^*$. If $\mathcal{A}$ is clear from the context, we shall omit it.*

The key observation is the following.

Theorem 1. *Given an application scenario $\mathcal{A} = \langle R_s, R_t, (\!|\cdot|\!)\rangle$, an incompleteness semantics $[\![\cdot]\!]^*$, and a query Q over R_t, if D is an instance of R_s and D' is a cover of D under $[\![\cdot]\!]^*$, then every correct answer to Q over $[\![D']\!]^*$ is a correct answer to Q over $\mathcal{A}(D)$.*

As our goal is to find correct answers to Q over $\mathcal{A}(D)$, this observation suggests looking for covers under some semantics of incompleteness and then providing correct answers for them. In the next section we show that such covers are very easy to find in the main applications we study here.

Running the query on a cover does not yet guarantee correctness, due to the presence of false positive, informally introduced earlier. For our purposes, it suffices to define them for queries returning sets of tuples. In the setting of Theorem 1, a tuple $\bar{a}$ is a *false positive* if it belongs to $Q(D')$ but is not a correct answer to Q over $\mathcal{A}(D)$.

Remark. While we chose to work with the notions of correctness and certainty based on relative informativeness of query answers, it is very common in the literature to define them as intersections of all query answers, i.e., $\Box_\cap(Q, \mathcal{A}, D) = \bigcap_{D' \in \mathcal{A}(D)} Q(D')$. Note that such answers contain only constant tuples, i.e., tuples without nulls. It turns out that our more general approach to certain and correct answers also provides guarantees for such answers.

Proposition 1. *Let $\mathcal{A} = \langle R_s, R_t, (\!|\cdot|\!)\rangle$ be an application scenario, Q a query over R_t, and D an instance of R_s. If D' is a cover of D under $[\![\cdot]\!]^*$, and $\bar{c}$ is a constant tuple that belongs to a correct answer to Q on D' under $[\![\cdot]\!]^*$, then $\bar{c} \in \Box_\cap(Q, \mathcal{A}, D)$.*

Thus, for the definition based on the intersection operator, to approximate query answers it still suffices to find a correct answer on a cover, and then restrict to constant tuples in it.

4 Correct Answers in Application Scenarios

We show how the approximation idea of Theorem 1 works in four application scenarios: data exchange, data integration, CQA, and OBDA. Specifically,

- finding correct answers to queries over incomplete databases under the CWA semantics works for closed-world data exchange, exact and complete data integration, and CQA;
- finding correct answers to queries over incomplete databases under the OWA semantics works for open-world data exchange, sound data integration, and OBDA.

In all these application scenarios, finding certain answers is computationally hard even for first-order queries, with data complexity ranging from coNP-hard to undecidable (Fagin et al. 2005; Lenzerini 2002; Bertossi 2011). However, by identifying suitable covers, we can find approximations efficiently. This section explains how covers can be constructed and gives general statements about the existence of approximations; then in Sections 5 and 6 we show concrete instances of reducing complexity from intractable to tractable.

Data Exchange

First recall some basic definitions, cf. (Arenas et al. 2014). A *schema mapping* $\mathcal{M}$ is a tuple $\langle R_s, R_t, \Sigma_{st}, \Sigma_t \rangle$, where R_s and R_t are source and target relational schemas, Σ_{st} is a set of source-to-target dependencies that specify how R_s instances are restructured under R_t, and Σ_t is a set of constraints over R_t. This is represented by an application scenario $\mathcal{A}_{\mathcal{M},*} = \langle R_s, R_t, (\!|\cdot|\!)^*_\Sigma \rangle$, where $\Sigma = \Sigma_{st} \cup \Sigma_t$, and, given a source instance $D \in \mathit{Inst}(R_s)$, $(\!|D|\!)^*_\Sigma$ is the set of *solutions* in data exchange under semantics indicated by $*$. Most commonly one uses an open-world (OWA) or closed-world (CWA) semantics. Under OWA, instances in $(\!|D|\!)^{\text{OWA}}_\Sigma$ are $D' \in \mathit{Inst}(R_t)$ such that D and D' together satisfy Σ_{st} and D' satisfies Σ_t. Under CWA, instances in $(\!|D|\!)^{\text{CWA}}_\Sigma$ are $D' \in \mathit{Inst}(R_t)$ satisfying Σ_t in which every fact is justified by D and Σ_{st} (see (Arenas et al. 2014; Hernich, Libkin, and Schweikardt 2011) for the somewhat involved definition of being justified). It is known that $(\!|D|\!)^{\text{CWA}}_\Sigma \subseteq (\!|D|\!)^{\text{OWA}}_\Sigma$. We shall refer to OWA- or CWA-data exchange, depending on the semantics of the solutions.

A special solution, called *canonical (universal) solution* for D and denoted by $\mathit{CanSol}_\mathcal{M}(D)$, plays a very important role in data exchange. It is constructed by applying constraints from Σ_{st} to D; these constraints are the form $\forall \bar{x}\ \phi(\bar{x}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})$, where ϕ and ψ are conjunctive queries over R_s and R_t. One example was given in Figure 1. If $\Sigma_t \neq \emptyset$, this is followed by applying the *chase* procedure, i.e., using constraints in Σ_t to derive new facts. Canonical universal solution is the preferred solution in data exchange, and is the one that is most often materialized for query answering. It is also a cover.

Proposition 2. *Let $\mathcal{M}$ be a schema mapping, and D a source instance. Assume that $\mathit{CanSol}_\mathcal{M}(D)$ exists. Then $\mathit{CanSol}_\mathcal{M}(D)$ is an $\mathcal{A}_{\mathcal{M},*}$-cover of D under $[\![\cdot]\!]^*$, where $*$ is* OWA *or* CWA.

Combining this with Theorem 1, we have a recipe for finding correct answers in data exchange for *arbitrary* queries: build the canonical solution $\mathit{CanSol}_\mathcal{M}(D)$, and apply any known algorithm for approximating certain answers with $\mathit{CanSol}_\mathcal{M}(D)$ viewed as an incomplete database. Note that we do not put any restrictions on the query Q.

Corollary 1. *If $\mathcal{M}$ is a schema mapping, D a source instance, Q an arbitrary query over R_t, and $\mathit{CanSol}_\mathcal{M}(D)$ exists, then every correct answer to Q over $[\![\mathit{CanSol}_\mathcal{M}(D)]\!]^*$ is a correct answer to Q over $\mathcal{A}_{\mathcal{M},*}(D)$, where $*$ is* OWA *or* CWA.

Sometimes this can be pushed further, and we can capture all correct answers. The treatment below applies in other application scenarios as well, but due to space limitations we only present full details for the case for data exchange.

For this, we restrict the class of queries, but still not nearly as much as UCQs, allowing a fair amount of negation in them. We use the class $\text{Pos}(\forall\text{G})$ of first-order formulae, defined in (Compton 1983). It contains the positive fragment, i.e., all atomic formulae, is closed under $\wedge, \vee, \exists, \forall$, and is also closed under the following *universal guarded* rule: if $\phi(\bar{x}, \bar{y})$ is a formula in $\text{Pos}(\forall\text{G})$, and $\alpha(\bar{x})$ is an atomic formula with the variables in $\bar{x}$ distinct, then $\forall \bar{x}\ (\alpha(\bar{x}) \rightarrow \phi(\bar{x}, \bar{y}))$ is a formula in $\text{Pos}(\forall\text{G})$. This is a very expressive fragment: it corresponds to relational algebra operations of selection, projection, union, join, and division by a relation (i.e., queries like 'find students who take all courses').

Proposition 3. *If $\mathcal{M}$ is a schema mapping, D a source instance, Q a query over R_t, and $T = \mathit{CanSol}_\mathcal{M}(D)$ exists, then:*

- $Q([\![T]\!]^{\text{OWA}}) \sim Q(\mathcal{A}_{\mathcal{M},\text{OWA}}(D))$ *for $Q \in$ UCQ;*
- $Q([\![T]\!]^{\text{CWA}}) \sim Q(\mathcal{A}_{\mathcal{M},\text{CWA}}(D))$ *for $Q \in \text{Pos}(\forall\text{G})$.*

Recall that the $\sim$ notation means that correct answers are the same over both families of databases (see Definition 1).

For queries from UCQ and $\text{Pos}(\forall\text{G})$, certain answers under OWA and CWA respectively can be found by *naively* evaluating Q over the incomplete database, i.e., treating nulls as distinct constants, cf. (Imielinski and Lipski 1984; Gheerbrant, Libkin, and Sirangelo 2014). This gives us the following.

Corollary 2. *Let $\mathcal{M}$ be a schema mapping, D a source instance, and Q a query over R_t. If $\mathit{CanSol}_\mathcal{M}(D)$ exists, then $Q(\mathit{CanSol}_\mathcal{M}(D))$ is the certain answer to Q on:*

1. *$\mathcal{A}_{\mathcal{M},\text{OWA}}(D)$ if $Q \in$ UCQ; and*
2. *$\mathcal{A}_{\mathcal{M},\text{CWA}}(D)$ if $Q \in \text{Pos}(\forall\text{G})$.*

This goes beyond known results (Fagin et al. 2005), which only proved item 1, and only for tuples without nulls.

Data Integration

A data integration system (Lenzerini 2002) is a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \Sigma \rangle$, where $\mathcal{G}$ is the global schema, $\mathcal{S}$ is the source schema (both assumed to be relational), and Σ is the mapping between them, consisting of rules of the form $q_S \rightsquigarrow q_G$ or $q_G \rightsquigarrow q_S$, where q_S is a k-ary conjunctive query over $\mathcal{S}$ and q_G is a k-ary conjunctive query over $\mathcal{G}$. More precisely, we talk about *LAV* (local as view) mappings for rules of the form $s \rightsquigarrow q_G$, where s is a k-ary predicate in $\mathcal{S}$, *GAV* (global as view) mappings for rules of the form $g \rightsquigarrow q_S$ where g is an k-ary predicate in $\mathcal{G}$, and *GLAV* mappings for arbitrary combinations of the previous two.

This is captured by an application scenario $\mathcal{A}_{\mathcal{I},*} = \langle \mathcal{S}, \mathcal{G}, (\!|\cdot|\!)^*_\Sigma \rangle$, where $(\!|\cdot|\!)^*_\Sigma$ produces instances of the global schema consistent with the source and the mapping, and $*$ is one of the standard assumptions on the interpretation of the mapping rules, such as *exact (e)*, *sound (s)* or *complete (c) view*. For a source instance $D \in \mathit{Inst}(\mathcal{S})$, a database D' over the global schema is in $(\!|D|\!)^*$ iff for each rule $q_S \rightsquigarrow q_G$ of Σ, the pair (D, D') satisfies $\forall \bar{x}\ q_S(\bar{x})\ \theta\ q_G(\bar{x})$, where θ is $\rightarrow$ for $* = s$, or $\leftarrow$ for $* = c$, or $\leftrightarrow$ for $* = e$.

Consider a LAV mapping. As for data exchange, we start with a source instance D and produce the canonical solution $\mathit{CanSol}_\mathcal{I}(D)$ by viewing mapping rules as source-to-target dependencies. This gives us covers for all the interpretations:

Proposition 4. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \Sigma \rangle$ be a data integration system and D a source instance. Then $\mathit{CanSol}_\mathcal{I}(D)$ is an $A_{\mathcal{I},*}$-cover of D under $[\![\cdot]\!]^{\text{OWA}}$ for $* = s$, and under $[\![\cdot]\!]^{\text{CWA}}$ for $* \in \{c, e\}$.*

Combined with Theorem 1, this gives us:

Corollary 3. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \Sigma \rangle$ be a data integration system, D a source instance, and Q an arbitrary query over $\mathcal{G}$. Then every correct answer to Q over $[\![\mathit{CanSol}_{\mathcal{I}}(D)]\!]^{\text{OWA}}$ is a correct answer to Q over $\mathcal{A}_{\mathcal{I},s}(D)$, and every correct answer to Q over $[\![\mathit{CanSol}_{\mathcal{I}}(D)]\!]^{\text{CWA}}$ is a correct answer to Q over $\mathcal{A}_{\mathcal{I},e}(D)$ and $\mathcal{A}_{\mathcal{I},c}(D)$.*

Consistent Query Answering (CQA)

In CQA applications, we have a database schema $\mathcal{R}$, a set of constraints Σ, and a database D that violates Σ. The problem is to find query answers one can be certain about even though the database violates constraints (Arenas, Bertossi, and Chomicki 1999; Bertossi 2011). For this, one defines the notion of a *repair* of D as a database D' that satisfies Σ and differs from D as little as possible. There are several ways of formalizing this by imposing a minimality criterion on the difference between D and D'. We shall look at a specific case where there is a single universally agreed notion of repair. This happens when constraints are given as keys – the most common database constraints (Abiteboul, Hull, and Vianu 1995) and in fact constraints most commonly studied in the context of CQA (Bertossi 2011).

Assume that each relation name $R \in \mathcal{R}$ has a *key*, i.e., a set of attributes K of R such that no two tuples of R have the same values of these attributes. If a relation violates the key constraint, then for each possible tuple $\bar{t}$ of values of K, it may have multiple tuples $\bar{t}_1, \ldots, \bar{t}_m$ that coincide with $\bar{t}$ on attributes in K. Then for each such a group a repair will pick just one tuple, to restore consistency (thus leading to potentially an exponential number of repairs). We let $\text{Rep}_\Sigma(D)$ be the set of all repairs of D. Consistent query answers then must be true in all repairs.

A CQA setting $\mathcal{C} = \langle \mathcal{R}, \Sigma \rangle$ is modeled as an application scenario $\mathcal{A}_{\mathcal{C}} = \langle \mathcal{R}, \mathcal{R}, (\!|\cdot|\!)_\Sigma \rangle$, where $(\!|D|\!)_\Sigma = \text{Rep}_\Sigma(D)$ for a database D. Unlike in the previous examples, we cannot rely on the canonical solution. Instead we use the *canonical repair* $\mathit{CanRep}_\Sigma(D)$, which is an incomplete database defined as follows. For each tuple $\bar{t}$ of values of K, if we have multiple tuples $\bar{t}_1, \ldots, \bar{t}_m$ that coincide with $\bar{t}$ on attributes in K, then we replace them with one new tuple $\bar{t}'$ that coincides with $\bar{t}$ on attributes in K, and has fresh nulls as values of other attributes.

Proposition 5. *Let $\mathcal{C} = \langle \mathcal{R}, \Sigma \rangle$ be a CQA setting where all constraints in Σ are keys, and D an instance over $\mathcal{R}$. Then $\mathit{CanRep}_\Sigma(D)$ is a cover of D under $[\![\cdot]\!]^{\text{CWA}}$.*

Again, combining with Theorem 1, we get:

Corollary 4. *Let $\mathcal{C} = \langle \mathcal{R}, \Sigma \rangle$ be a CQA setting where Σ contains keys, D an instance over $\mathcal{R}$ and Q an arbitrary query over R. Then every correct answer to Q over $[\![\mathit{CanRep}_\Sigma(D)]\!]^{\text{CWA}}$ is a correct answer to Q over $\mathcal{A}_{\mathcal{C}}(D)$.*

Ontology-Based Data Access

We recall that an *ontology* is constituted by a pair $\langle \mathcal{T}, \mathcal{A} \rangle$. where $\mathcal{T}$ is called TBox and contains the intensional knowledge of the ontology, i.e., the inference rules, while $\mathcal{A}$ is called ABox and contains the extensional part of the ontology, i.e., the facts. By $\mathit{sgn}(\mathcal{T})$ we denote the signature of $\mathcal{T}$, i.e., the set of predicates occurring in $\mathcal{T}$.

An *OBDA specification* (Poggi et al. 2008) is a tuple $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ consisting of a relational schema $\mathcal{R}$, a TBox $\mathcal{T}$, and a set of mappings $\mathcal{M}$ between the predicates of $\mathcal{T}$ and the predicates of $\mathcal{R}$. In this context, we assume $\mathcal{M}$ to be a set of GAV mapping (as defined earlier) of the form $g \rightsquigarrow q_g$, where g is a k-ary predicate in $\mathit{sgn}(\mathcal{T})$ and q_g is a k-ary conjunctive query over $\mathcal{R}$. Then for a database D of $\mathcal{R}$, by $\mathcal{M}(D)$ we mean the database of $\mathit{sgn}(\mathcal{T})$ in which each relation g is interpreted as $q_g(D)$.

Given a database instance D over $\mathcal{R}$, such a specification denotes the set of all the ABoxes that can be obtained by translating the facts in D in terms of $\mathit{sgn}(\mathcal{T})$ using the rules in $\mathcal{M}$, and extended with the knowledge that can be inferred from $\mathcal{T}$. This is modeled by an application scenario $\mathcal{A}_{\mathcal{O}} = \langle \mathcal{R}, \mathit{sgn}(\mathcal{T}), (\!|\cdot|\!)_\Sigma \rangle$, where $\Sigma = \mathcal{T} \cup \mathcal{M}$, and $(\!|D|\!)_\Sigma = \{D' \mid \mathcal{M}(D) \subseteq D' \text{ and } D' \models \mathcal{T}\}$, see (Poggi et al. 2008; Bienvenu and Ortiz 2015).

This also applies to a simplified OBDA scenario, known as *ontology-mediated query answering* (OMQA), in which an actual ABox over $\mathit{sgn}(\mathcal{T})$ takes the place of the virtual ABox constituted by $\langle D, \mathcal{M} \rangle$ in the above description.

Similarly, given an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, such a case denotes the set of all ABoxes that constitute extensions of $\mathcal{A}$ and satisfy the ontology constraints, and can be also modeled by an application scenario $\mathcal{A}_{\mathcal{O}} = \langle \mathit{sgn}(\mathcal{T}), \mathit{sgn}(\mathcal{T}), (\!|\cdot|\!)_{\mathcal{T}} \rangle$, where $(\!|\mathcal{A}|\!)_{\mathcal{T}} = \{\mathcal{A}' \supseteq \mathcal{A} \mid \mathcal{A}' \models \mathcal{T}\}$.

The complexity of OBDA query answering has been studied almost exclusively for CQs and their unions. The few exceptions one can find in the literature, in fact, have the purpose of showing the limitations of ontology languages beyond such well-behaved classes (Rosati 2007; Gutiérrez-Basulto et al. 2015). Techniques for finding certain answers generally follow three approaches. One is by rewriting the original query in a tractable language (e.g., FO) to obtain the certain answers, as is done, e.g., for ontologies in DL-Lite (Calvanese et al. 2007), some fragments of Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Pieris 2012), or weakly recursive dependencies (Civili and Rosati 2012). Another approach is to use the *chase* procedure that uses ontology rules to derive new facts, and apply the query on its result, if the procedure terminates. This includes ontologies with weakly acyclic tgds (Fagin et al. 2005) or acyclic graphs of rule dependencies (Baget et al. 2011), where chase always terminates, or guarded Datalog$^{\pm}$ (Calautti, Gottlob, and Pieris 2015), where termination is decidable. The third combined approach, often used with the $\mathcal{EL}$ family of description logics (Baader, Brandt, and Lutz 2005; Lutz, Toman, and Wolter 2009), uses both chase and rewriting.

Here we look at the approach based on the chase procedure, as defined, in the case of ontological languages mentioned above, in (Greco, Molinaro, and Spezzano 2012) and other references above. We denote the result of its application, if it terminates successfully, by $\mathit{chase}(\mathcal{M}(D), \mathcal{T})$. Then:

Proposition 6. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{M} \rangle$ be an OBDA specification and D a relational instance over $\mathcal{R}$. If $\mathit{chase}(\mathcal{M}(D), \mathcal{T})$ is well-defined (i.e., chase terminates without failures), then it is an $\mathcal{A}_{\mathcal{O}}$-cover of D under $[\![\cdot]\!]^{\text{OWA}}$.*

Once again, from Theorem 1, we derive the following.

Corollary 5. *Let $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{M} \rangle$ be an OBDA specification, D a relational instance over $\mathcal{R}$ and Q an arbitrary query over sgn($\mathcal{T}$). If chase($\mathcal{M}(D), \mathcal{T}$) is well-defined (i.e., chase terminates without failures) then every correct answer to Q over $[\![chase(\mathcal{M}(D), \mathcal{T})]\!]^{\text{OWA}}$ is a correct answer to Q over $\mathcal{A}_{\mathcal{O}}(D)$.*

Notice that, in the simplified scenario of OMQA mentioned earlier, the above results still apply. In particular, given an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, we have that $chase(\mathcal{A}, \mathcal{T})$ is an $\mathcal{A}_{\mathcal{O}}$-cover of $\mathcal{A}$ under $[\![\cdot]\!]^{\text{OWA}}$, therefore, every correct answer to an arbitrary query Q over $[\![chase(A, \mathcal{T})]\!]^{\text{OWA}}$ is a correct answer to Q over $\mathcal{A}_{\mathcal{O}}(\mathcal{A})$.

5 Approximations under Closed World

All the results of the previous section were of the same shape: correct answers over a particular cover, under some semantics of incompleteness, are also correct answers in application scenarios. For the former, we know (at least in some cases), how to find nontrivial correct answers efficiently even if certain answers are intractable. Such results exist primarily under CWA, where finding certain answers is coNP-complete, as opposed to OWA, where certain answers cannot be computed for arbitrary FO queries (Abiteboul, Kanellakis, and Grahne 1991). Indeed, it is easier to approximate a coNP-complete rather than an undecidable problem. We now look at one existing approximation scheme for queries over incomplete data, and show that it lets us produce nontrivial approximations of query answering in CWA data exchange, complete or exact data integration, and CQA scenarios (these are exactly the scenarios from Section 4 where the CWA semantics of incompleteness was used).

As a concrete approximation scheme, we use one from (Guagliardo and Libkin 2016). It takes a first-order query Q and translates it into another first-order query Q^+ with the following properties:

1. $Q^+(D) \subseteq \Box_\bot(Q, D)$ for every database D, and thus $Q^+(D)$ is a correct answer on $[\![D]\!]^{\text{CWA}}$;
2. $Q^+(D) = Q(D)$ if D is complete (i.e., it does not miss any answers in the case there are no nulls in the database); and
3. if Q is expressed in relational algebra (as, e.g., the query in the introduction), then Q^+ is linear in the size of Q.

This, together with the results of the previous section, implies that if, in an application scenario $\mathcal{A}$, an instance D' is a cover, under $[\![\cdot]\!]^{\text{CWA}}$, for a source database D, then $Q^+(D')$ is a correct answer to Q over $\mathcal{A}(D)$.

We now illustrate how this approximation scheme will work for Example 1.

Example 2. We first formalize the setting of Example 1 as an application scenario. Let $\mathcal{A}_{\mathcal{M},\text{CWA}}$ be $\langle R_s, R_t, (\!|\cdot|\!)^{\text{CWA}}_{\Sigma} \rangle$ with R_s, R_t and Σ as in Example 1. The canonical solution $\mathit{CanSol}_{\mathcal{M}}(D)$ is shown in Figure 1. From Theorem 1, we know that every correct answer to Q over $[\![\mathit{CanSol}_{\mathcal{M}}(D)]\!]^{\text{CWA}}$ is also a correct answer to Q over $\mathcal{A}_{\mathcal{M},\text{CWA}}(D)$. Using the transformation of (Guagliardo and Libkin 2016) on Q, we obtain Q^+, again expressed in relational algebra:

$$\begin{array}{ll} & \pi_{\mathsf{oid}}(\mathsf{O} - (\mathsf{O} \ltimes_{\text{unif}} \pi_{\mathsf{oid,cid,total}}(\mathsf{O} \bowtie_{\mathsf{oid=poid}} \mathsf{P})) \\ \cup & \pi_{\mathsf{oid}}(\sigma_{\mathsf{pdate} \geq \$\mathsf{today} \wedge \mathsf{const(pdate)}}(\mathsf{O} \bowtie_{\mathsf{oid=poid}} \mathsf{P})) \end{array}$$

Here $\ltimes_{\text{unif}}$ is the left unification semijoin, i.e.,

$$R \ltimes_{\text{unif}} S = \{\bar{r} \in R \mid \exists \bar{s} \in S : \text{unif}(\bar{r}, \bar{s})\}.$$

The condition unif$(\bar{r}, \bar{s})$ says that $\bar{r}$ and $\bar{s}$ are unifiable, i.e., $h(\bar{r}) = h(\bar{s})$ for some homomorphism h. This is checkable in linear time (Paterson and Wegman 1978).

Recall that $Q(\mathit{CanSol}_{\mathcal{M}}(D))$ was not a correct answer, as it produced two false positives, 02 and 03. But when Q^+ is evaluated over $\mathit{CanSol}_{\mathcal{M}}(D)$, it produces a single correct answer 01, eliminating false positives. □

We know that in all the application scenarios where covers under the CWA semantics are constructed – closed-world data exchange, exact and complete data integration, CQA – the complexity of answering FO queries could be coNP-complete (Arenas et al. 2014; Hernich, Libkin, and Schweikardt 2011; Bertossi 2011). The complexity of evaluating Q^+ is polynomial (even logspace), as it is an FO query. Thus, using Q^+ over the cover in these scenarios does seem to reduce the complexity of query answering from coNP to tractable.

Nontrivial approximations There are several situations when going through the procedure we described – that is, constructing a cover and answering Q^+ over it – produces effectively a trivial approximation. For example, if Q^+ returns the empty set of tuples, it is hardly useful as an approximation for a coNP-hard query answering problem. Also, if D' is a cover, and $Q(D')$ produces no false positives, then one could argue that just running Q would already give useful information and computing Q^+ is an overkill. Also, if $Q(D')$ is the empty set of tuples, there is no need to approximate it using Q^+.

Thus, to demonstrate real usefulness of approximations, we need to show that they can be produced in cases where:

- evaluating Q directly on the cover gives false positives, or empty answers, and
- evaluating Q^+ on the cover returns some tuples.

In what follows, we formalize this idea by introducing the notion of an application scenario admitting a non-trivial approximation to a $\mathcal{C}$-hard $\mathcal{L}$-query answering problem, where $\mathcal{C}$ is a complexity class and $\mathcal{L}$ is a query language. The definition below essentially excludes the informally presented cases when approximation is not useful.

Definition 4. *An application scenario $\mathcal{A}$ admits a nontrivial approximation to a $\mathcal{C}$-hard $\mathcal{L}$-query answering problem if there exists a family of source databases $\mathcal{D} = \{D_i\}_{i \in \mathbb{N}}$ and a query $Q \in \mathcal{L}$ such that:*

a) the data complexity of computing $\Box_\cap(Q, \mathcal{A}, D)$ is $\mathcal{C}$-hard for instances $D \in \mathcal{D}$;

b) for each i, the answer $Q(D_i)$ is not useful for query answering in $\mathcal{A}$; that is, either $Q(D_i)$ has false positives or $Q(D_i) = \emptyset$;

c) there exist a query Q' that depends on Q and has polynomial time data complexity, such that for each i there is an $\mathcal{A}$-cover D'_i of D_i computable in polynomial time and $Q'(D'_i)$ is not empty and provides a correct answer to Q over $\mathcal{A}(D_i)$.

We now show that this is indeed the case for the three application scenarios mentioned above.

Theorem 2. *Each of the following application scenarios $\mathcal{A}$:*

1. CWA *data exchange,*
2. *complete or exact data integration,*
3. *CQA,*

admits a non-trivial approximation to a CONP*-hard FO-query answering problem.*

In all the three cases of Theorem 2, we find an FO query Q for which the problem of computing query answers is CONP-hard and the evaluation of Q over the cover produces false positives, yet the evaluation of the approximation Q^+ over the cover produces some tuples, all of which are correct answers.

6 Approximations under Open World

As we have seen in the previous section, the complexity of answering FO queries can already be CONP-complete for the cases in which we build covers under the CWA semantics. For the cases in which we build covers under the OWA semantics – open world data exchange, sound data integration, OBDA – the situation is even worse. In fact, it is well-known that computing certain answers to arbitrary FO queries under OWA is undecidable, in both data complexity and query complexity. That is (assuming for the simplicity of notation that Q is a Boolean FO query), checking whether $Q(\llbracket D \rrbracket^{\text{OWA}}) = \{\top\}$ could be undecidable even for a fixed FO query Q, and it likewise can be undecidable even for a fixed database D. Here of course $\top$ stands for true.

Therefore, there is little hope of getting a good approximation schema for the OWA semantics similar to the one in (Guagliardo and Libkin 2016); in fact no such scheme is known. One can then ask what are the advantages of building a cover under the OWA-semantics in the above mentioned cases. Although the situation looks more daunting than in the previous case, there are still some possibilities to exploit. One way is to restrict to classes of queries for which computing certain answers to arbitrary FO queries under the OWA is decidable, but still to classes that extend UCQs.

One option is to look at prefix-vocabulary classes, for which the classical decision problem has been thoroughly investigated (Börger, Grädel, and Gurevich 2001), and concentrate on decidable classes with the finite model property, so that our results would be applicable in the finite case. The most fitting such class is the $\exists^*\forall^*$ prefix class, known as the Bernays-Schönfinkel class. Its formulae are those whose quantifier prefix consists of existential quantifiers, followed by universal quantifiers, followed by a quantifier-free formula. Formulae in this class have the finite model property, and their satisfiability problem has NEXPTIME complexity, or PSPACE complexity for a fixed relational vocabulary.

The idea of reducing certain answers to satisfiability is as follows. Given an incomplete database D with k constants and m nulls, we can write a formula ϕ_D with the existential prefix $\exists y_1 \dots \exists y_k \exists x_1 \dots \exists x_m$ whose quantifier-free part has the conjunction of all inequalities $y_i \neq y_j$ for $i \neq j$ and of all the facts in D, with variables $y_1, \dots, y_k$ used in place of constants and $x_1, \dots, x_m$ used in place of nulls. Then finite models of this formula are precisely the databases isomorphic to databases in the OWA semantics of D.

Now given a Boolean query Q, consider $\beta(Q, D) = \phi_D \to Q$. Then $Q(\llbracket D \rrbracket^{\text{OWA}}) = \{\top\}$ iff $\beta(Q, D)$ is finitely valid (since we only deal with finite validity and satisfiability, we omit the word finite from now on). This happens iff $\neg\beta(Q, D)$ is not satisfiable. Thus, to compute certain answers under OWA, we need to check satisfiability of $\beta(Q, D)$. Since ϕ_D is an existential formula, it follows that if Q is in the $\forall^*\exists^*$ class, then $\beta(Q, D)$ is in the Bernays–Schönfinkel class. This gives us the following.

Proposition 7. *Let $\mathcal{R}$ be a relational schema, D an incomplete database and Q a boolean FO query over $\mathcal{R}$ which belongs to the $\forall^*\exists^*$ class. Then $Q(\llbracket D \rrbracket^{\text{OWA}}) = \{\top\}$ can be checked in coNEXPTIME, or in PSPACE if $\mathcal{R}$ is fixed.*

There are several well known classes of queries which satisfy conditions of Proposition 7:

- UCQs with inequalities, denoted by $\text{UCQ}^{\neq}$, are FO formula of the form $Q_1 \vee \dots \vee Q_n$ where each Q_i is a CQ extended with inequality atoms $x \neq y$ (for example, $\exists y, z\ P(x, y) \wedge P(y, z) \wedge y \neq z$);
- Boolean combinations of conjunctive queries (BCCQs) are obtained from CQs by closing them under union, intersection and difference.

Corollary 6. *For each of the following application scenarios $\mathcal{A}$: (i)* OWA *data exchange, (ii) sound data integration, (iii) OBDA, given a BCCQs or a $\text{UCQ}^{\neq}$ query Q and an instance D, if D' is an $\mathcal{A}$-cover D' of D under $\llbracket\cdot\rrbracket^{\text{OWA}}$, then $Q(\llbracket D' \rrbracket^{\text{OWA}})$ is a correct answer to Q over $\mathcal{A}(D)$, computable with coNEXPTIME combined and PSPACE data complexity in the sizes of Q and D'.*

The literature on application scenarios captured by OWA covers uses almost entirely UCQs as queries to be answered, with very few exceptions, e.g., (Arenas, Pérez, and Reutter 2013), that provide very mild extensions. Corollary 6 gives a way of finding approximations to queries that extend UCQs in more significant ways, allowing a fair amount of negation. We now investigate their complexity in more details and show under which circumstances our approach generates nontrivial approximations of reasonable complexity.

Boolean Combinations of Conjunctive Queries For the case of BCCQs, we show how to reduce the prohibitively expensive combined complexity (coNEXPTIME, effectively double-exponential) to single-exponential. More specifically, we show that, given an application scenario $\mathcal{A}$ and a source instance D for which there exists an $\mathcal{A}$-cover under $\llbracket\cdot\rrbracket^{\text{OWA}}$, the complexity of checking certainty of a BCCQ in such a scenario drops to EXPTIME.

Theorem 3. *Let $\mathcal{A} = \langle R_s, R_t, (\!|\cdot|\!)_\Sigma \rangle$ be one of the following application scenarios:*

1. OWA *data exchange,*
2. *sound data integration, or*
3. *OBDA.*

Let D be a source instance, and Q a BCCQ. If there exists an $\mathcal{A}$-cover D' of D under $[\![\cdot]\!]^{\text{OWA}}$, then the problem of checking whether $Q(\mathcal{A}(D)) = \{\top\}$ is in EXPTIME in combined complexity and PSPACE in data complexity.

This result is based on a lemma showing that, in each such application scenarios, for two conjunctive queries Q_1 and Q_2, the problem of checking whether $Q_1 \subseteq Q_2$ on all databases in $\mathcal{A}(D)$ is in EXPTIME. Checking certainty of a BCCQ is then reduced to such containment checking.

UCQs with inequalities From Section 4, we know that if $\mathcal{A}$ in an OWA application scenario as one of those listed in Theorem 3, D an instance, D' is an $\mathcal{A}$-cover of D under $[\![\cdot]\!]^{\text{OWA}}$, and Q an arbitrary query, then every correct answer to Q on D' under OWA is a correct answer to Q on $\mathcal{A}(D)$. In particular, if we have any table R such that $R \preceq Q([\![D']\!]^{\text{OWA}})$, then such R is a correct answer in the application scenario.

We now show how to find such an R when Q is a UCQ$^{\neq}$ query. Of course Theorem 7 tells us that we can check for satisfiability of $\neg\beta(Q, D')$, but this is rather expensive, PSPACE in data complexity and coNEXPTIME in combined complexity. Instead, we propose a different procedure of much lower complexity.

Let Q be a UCQ$^{\neq}$ query over schema $\mathcal{R}$. We define a new query $Q^{\neq}$, over $\mathcal{R}$ expanded with a unary predicate IsNull interpreted as follows: $\mathsf{IsNull}(a)$ is true iff $a \in \mathsf{N}$. This corresponds to the standard $\mathsf{IS\ NULL}$ statement found in query languages such as SQL. Now $Q^{\neq}$ is obtained by replacing each atom $x \neq y$ with

$$(x \neq y) \wedge \neg\,\mathsf{IsNull}(x) \wedge \neg\,\mathsf{IsNull}(y)\,.$$

Lemma 1. *Let D be an incomplete database and Q a UCQ$^{\neq}$. Then $Q^{\neq}(D) \preceq Q([\![D]\!]^{\text{OWA}})$, i.e., $Q^{\neq}(D)$ is a correct answer to Q on D under* OWA.

Combined with previous results, this gives us an approach for approximating unions of conjunctive queries with negation in application scenarios.

Theorem 4. *Let $\mathcal{A} = \langle R_s, R_t, (\!|\cdot|\!)_\Sigma\rangle$ be one of the following application scenarios $\mathcal{A}$:*

1. OWA *data exchange,*
2. *sound data integration,*
3. *OBDA.*

Let D be a source instance, Q a UCQ$^{\neq}$ query, and D' an $\mathcal{A}$-cover of D under $(\!|\cdot|\!)^{\text{OWA}}$. Then $Q^{\neq}(D')$ is a correct answer to Q on $\mathcal{A}(D)$.

To make this result applicable in OWA data exchange, and OBDA, we need to find suitable covers (it can be used in data integration as well, but we omit additional technical details due to space limitations). For these scenarios, as discussed in Section 4, results of terminating chase sequences can be used as covers. However, quite often, a terminating chase is hard to achieve without strong restrictions. But in our case there is a way around it: one can chase for a finite number of steps, and the result is still a cover under our definition. Such a cover of course would only be useful if running $Q^{\neq}$ over it produces some tuples (which are then guaranteed to be correct answers). We now show that this can happen, and chasing for a finite number of steps could be useful for answering UCQ$^{\neq}$ queries, if chase does not terminate.

Lemma 2. *Consider* OWA *data exchange and OBDA application scenarios and a source instance D. Then, for every $k > 0$, chasing D_0 with constraints in C is a cover, where:*

- *for data exchange, $D_0 = D$ and $C = \Sigma$;*
- *for OBDA, D_0 is the result of applying the GAV mapping $\mathcal{M}$ to D, and $C = \mathcal{T}$.*

The above result is useful towards the goal of approximating UCQ$^{\neq}$. We recall that the complexity of handling query answering with inequalities for the application scenarios of interest for this section ranges from coNP to undecidable, depending on the constraint language used (Rosati 2007; Gutiérrez-Basulto et al. 2015; Abiteboul and Duschka 1998). Building on these results, and mirroring what we did in Section 5, we conclude this section by showing the application scenarios discussed in this section admit a non-trivial approximations (using Definition 4 for the concept of non-trivial approximations).

Theorem 5. *Both* OWA *data exchange and OBDA application scenarios admit non-trivial approximations to a* coNP*-hard UCQ$^{\neq}$-query answering problem.*

Specifically, we show how to find, for each scenario $\mathcal{A}$, source instances D and a UCQ$^{\neq}$ query Q so that finding certain answer to Q over $\mathcal{A}(D)$ is coNP-hard. Then, using Lemma 2, we construct in polynomial time the new instance D_0. Running $Q^{\neq}$ on D_0 results in the empty table and thus is not useful. However, we can chase D_0 for just three steps in order to get a new instance D_3 such that $Q^{\neq}(D_3)$ is not empty – and thus returns tuples which are correct answers.

7 Conclusions

We proposed a framework aimed at generalizing multiple scenarios in which one or more source databases are enhanced with an additional layer of knowledge that is taken into account for query answering. We showed how query answering can be approximated via producing query answers over incomplete database instances. When such approximations work under closed-world assumption, we obtain efficient approximations of coNP-hard query answering problems. Under OWA, we can still go beyond UCQs, although not as far as in the case of CWA.

Thus, as the main future direction, we aim at extending the classes of queries that this approach can capture via the OWA semantics of incompleteness. We would like to devise new approximation schemes that capture larger classes of FO than extensions of UCQs with inequalities and Boolean combinations.

Acknowledgments We thank referees for their helpful comments. Work supported by EPSRC grants M025268 and N023056.

References

Abiteboul, S., and Duschka, O. M. 1998. Complexity of answering queries using materialized views. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 254–263.

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Abiteboul, S.; Kanellakis, P.; and Grahne, G. 1991. On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78(1):158–187.

Arenas, M.; Barceló, P.; and Reutter, J. L. 2011. Query languages for data exchange: Beyond unions of conjunctive queries. *Theory Comput. Syst.* 49(2):489–564.

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *PODS*, 68–79.

Arenas, M.; Barceló, P.; Libkin, L.; and Murlak, F. 2014. *Foundations of Data Exchange*. Cambridge University Press.

Arenas, M.; Pérez, J.; and Reutter, J. L. 2013. Data exchange beyond complete data. *Journal of the ACM* 60(4).

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In *IJCAI*, 364–369.

Baget, J.; Leclère, M.; Mugnier, M.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artif. Intell.* 175(9-10):1620–1654.

Bertossi, L. 2011. *Database Repairing and Consistent Query Answering*. Morgan&Claypool Publishers.

Bienvenu, M., and Ortiz, M. 2015. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, 218–307.

Börger, E.; Grädel, E.; and Gurevich, Y. 2001. *The classical decision problem*. Springer Science & Business Media.

Calautti, M.; Gottlob, G.; and Pieris, A. 2015. Chase termination for guarded existential rules. In *PODS*, 91–103.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14:57–83.

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193:87–128.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning* 39(3):385–429.

Civili, C., and Rosati, R. 2012. A broad class of first-order rewritable tuple-generating dependencies. In *Datalog 2.0*, 68–80.

Compton, K. 1983. Some useful preservation theorems. *Journal of Symbolic Logic* 48(2):427–440.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336(1):89–124.

Gheerbrant, A.; Libkin, L.; and Sirangelo, C. 2014. Naïve evaluation of queries over incomplete databases. *ACM Trans. Database Syst.* 39(4):31:1–31:42.

Greco, S.; Molinaro, C.; and Spezzano, F. 2012. *Incomplete Data and Data Dependencies in Relational Databases*. Morgan & Claypool Publishers.

Greco, S.; Molinaro, C.; and Trubitsyna, I. 2017. Computing approximate certain answers over incomplete databases. In *AMW*.

Guagliardo, P., and Libkin, L. 2016. Making SQL queries correct on incomplete databases: A feasibility study. In *PODS*, 211–223.

Gutiérrez-Basulto, V.; Ibáñez-García, Y. A.; Kontchakov, R.; and Kostylev, E. V. 2015. Queries with negation and inequalities over lightweight ontologies. *J. Web Sem.* 35:184–202.

Hernich, A.; Libkin, L.; and Schweikardt, N. 2011. Closed world data exchange. *ACM Trans. Database Syst.* 36(2):14.

Imielinski, T., and Lipski, W. 1984. Incomplete information in relational databases. *Journal of the ACM* 31(4):761–791.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *PODS*, 233–246.

Libkin, L. 2016. Certain answers as objects and knowledge. *Artif. Intell.* 232:1–19.

Lipski, W. 1984. On relational algebra with marked nulls. In *PODS*, 201–203.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic EL using a relational database system. In *IJCAI*, 2070–2075.

Paterson, M., and Wegman, M. N. 1978. Linear unification. *J. Comput. Syst. Sci.* 16(2):158–167.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. Data Semantics* 10:133–173.

Reiter, R. 1977. On closed world data bases. In *Logic and Data Bases*, 55–76.

Rosati, R. 2007. The limits of querying ontologies. In *ICDT: International Conference on Database Theory*, 164–178.